

Máster Avanzado de Programación en Python para Hacking, BigData y Machine Learning

CERTIFICACIÓN PCAP

LECCIÓN 3

Listas, tuplas y diccionarios

ÍNDICE

- ✓ Introducción
- ✓ Objetivos
- ✓ Listas
- ✓ Tuplas
- ✓ Diccionarios
- ✓ Iterar a través de las listas, tuplas y diccionarios
- ✓ Conclusiones

INTRODUCCIÓN

En esta lección vamos a continuar con el repaso de los aspectos más relevantes del contenido cubierto por el PCAP. Para ellos veremos las principales estructuras de almacenamiento de datos: las listas, tuplas y diccionarios.

OBJETIVOS

Al finalizar esta lección serás capaz de:

- 1 Conocer las principales estructuras de almacenamiento de datos: listas, tuplas y diccionarios
- 2 Conocer sus características y métodos
- 3 Conocer los conceptos de objetos mutables e inmutables así como sus implicaciones
- 4 Conocer como iterar a través de las listas, tuplas y diccionarios

Listas: Definición

Una lista en Python es una colección de datos ordenada.



Importante

- Los valores pueden ser de cualquier tipo de dato, por lo que una misma lista podría contener enteros, flotantes, cadenas,...

```
mi_lista = [valor1, valor2, valor3,...]
```

Listas: Longitud de una lista



Importante

- La función `len()` devuelve un entero.

```
mi_lista = [valor1, valor2, valor3,...]  
len(mi_lista)|
```

Listas: Indexación



Importante

- En Python el primer elemento está en la posición 0.
- El índice debe ser un entero o booleano (se convierte en entero). Si flotante TypeError.
- Los índices pueden ser negativos. -1 es el último elemento de la lista.
- Si se intenta acceder a un índice que no existe IndexError.

```
mi_lista = [valor1, valor2, valor3,...]  
elemento = mi_lista[valor]
```


Listas: Añadir elementos al final de la lista



Importante

- Al utilizar la función añade el elemento al final de la lista.

```
mi_lista = [valor1, valor2, valor3]  
mi_lista.append(valor4)
```

Listas: Eliminar elementos de una lista



Importante

- Al eliminar un elemento modificamos la longitud de la lista y los índices de los elementos posteriores.
- Si intentamos eliminar un elemento de un índice que no existe IndexError.
- Si eliminamos la lista entera ya no existirá la variable NameError.

```
mi_lista = [valor1, valor2, valor3,...]  
del[indice]  
  
del mi_lista
```

Listas: Seleccionando elementos de una lista (Slices)



Importante

- El segundo índice indica el primero que NO se incluirá.
- El inicio y fin pueden estar vacíos. El salto es opcional (por defecto 1).
- Los valores de inicio, fin y salto pueden ser positivos o negativos.
- Los slices se pueden utilizar también para eliminar elementos.

```
mi_lista = [valor1, valor2, valor3,...]  
mi_lista[inicio:fin:salto]  
  
del mi_lista[inicio:fin:salto]
```

Listas: Mutabilidad de los objetos de tipo lista

Python es un lenguaje donde todo son “objetos”.

Los objetos pueden ser divididos en:

- Objetos inmutables (no se pueden modificar): enteros, cadenas, tuplas,...
- Objetos mutables (sí se pueden modificar): Listas y diccionarios

Función `id()` devuelve la dirección de memoria.



Importante

- Al intentar modificar un objeto inmutable, se crea un nuevo objeto.
- Al intentar modificar un objeto mutable, no se crea un objeto nuevo, se modifica el existente y todas las variables que apunten al objeto mostrarán ese cambio.

Tuplas: Definición

Una tupla en Python es un objeto inmutable que permite almacenar valores de manera ordenada.



Importante

- Si se intentan modificar devuelven `TypeError`.
- Se puede eliminar la tupla completa con `del` igual que en las listas.
- La indexación es igual que en las listas.

```
tupla1 = (valor1, valor2, ...)  
tupla2 = valor1, valor2, ...  
  
tupla1 = (valor1,)  
tupla2 = valor1,
```

Diccionarios: Definición

Un diccionario es un objeto mutable ordenado desde Python 3.7 formado por un conjunto de pares “clave-valor”.



Importante

- Los diccionarios no pueden tener claves repetidas.
- Las claves y valores pueden ser de cualquier tipo.
- Para conocer su longitud se puede usar el método len().

```
diccionario = {clave1: valor1,  
               clave2: valor2,  
               ...}
```

Diccionarios: Lectura y modificación de diccionarios



Importante

- Para acceder a los valores se debe acceder por la clave.
- Si la clave no existe devolverá KeyError.

```
diccionario = {clave1: valor1, clave2: valor2}
variable = diccionario[clave1]

diccionario[clave1] = nuevo_valor
diccionario[clave2] = nuevo_valor

del diccionario[clave1]
```

Diccionarios: Lectura y modificación de diccionarios



Importante

- `keys()` -> Lista con todas las claves (`dict_keys`).
- `values()` -> Lista con los valores (`dict_values`)
- `items()` -> Listas de tuplas con clave y valor (`dict_items`)

```
diccionario = {clave1: valor1, clave2: valor2}

diccionario.keys()
diccionario.values()
diccionario.items()
```


Iterar a través de las Listas, Tuplas y Diccionarios

Iterar a través de Listas

```
mi_lista = [valor1, valor2]

for elemento in mi_lista:
    hacer
```

Iterar a través de Tuplas

```
mi_tupla = (valor1, valor2)

for elemento in mi_tupla:
    hacer
```

Iterar a través de Diccionarios

```
mi_dict = {clave1: valor1,
            clave2: valor2}

for clave in mi_dict:
    hacer

for clave, valor in mi_dict.items():
    hacer
```

CONCLUSIONES

1

Hemos estudiado las estructuras principales de almacenamiento: listas, tuplas y diccionarios.

2

Hemos explicado las principales características de cada una, así como los procedimientos para interactuar con ellas.

3

Hemos visto los conceptos de objetos mutables e inmutables así como sus implicaciones.



MUCHAS GRACIAS POR SU ATENCIÓN



tcivera@grupomainjobs.com



Tamara Civera Lorenzo
es.linkedin.com/in/tamara-civera-lorenzo-95962147



twitter.com/eiposgrados



facebook.com/eiposgrados



instagram.com/eiposgrados